Instituto Tecnológico de Costa Rica

Escuela de Ingeniería en Computación

Curso: Bases de Datos I

Profesor: Ing. Franco Quirós Ramírez

Segunda Tarea Programada: Gestión de Empleados y Vacaciones

Estudiantes:

Andrés Baldi Mora

Elías Ramírez Hernández

II Semestre, 2025

Índice de Contenidos

Índice de Figuras

Capítulo 1: Introducción

En este documento se presenta la documentación técnica y los resultados obtenidos en la segunda tarea programada del curso de Bases de Datos I. El objetivo principal de este informe es demostrar el cumplimiento de los requerimientos funcionales y no funcionales, detallar el ambiente de desarrollo utilizado, y presentar las métricas relevantes del proyecto.

El proyecto consiste en una aplicación web diseñada para la gestión de empleados y el control de sus vacaciones. La solución implementa un CRUD completo para la tabla de empleados, un sistema de autenticación de usuarios (login/logout), el registro de movimientos de vacaciones (créditos y débitos), y una bitácora detallada para la trazabilidad de todas las operaciones. La arquitectura sigue un modelo de tres capas (presentación, lógica y datos), asegurando que toda la interacción con la base de datos se realice exclusivamente a través de procedimientos almacenados para maximizar la seguridad y la integridad de los datos.

A lo largo de este documento, se desglosarán los aspectos técnicos de la implementación, se evaluará el

grado de cumplimiento de cada requisito solicitado y se proporcionarán métricas cuantitativas sobre el esfuerzo y el alcance del desarrollo.

Capítulo 2: Contexto del Desarrollo

Descripción del Ambiente de Desarrollo

Para la realización de este proyecto se empleó un conjunto de herramientas colaborativas y tecnologías específicas para cada capa de la aplicación.

- **Control de Versiones:** Se utilizó **Git** y **GitHub** para gestionar el código fuente, permitiendo un trabajo colaborativo y un seguimiento detallado de los cambios. El repositorio del proyecto se encuentra en: https://github.com/EliPoli64/tarea2basesdatos.
- **Entorno de Desarrollo Integrado (IDE):** Se utilizó **Visual Studio Code** como editor de código principal para el desarrollo tanto del frontend como del backend.
- **Gestión de Base de Datos:** Se utilizó **SQL Server Management Studio (SSMS)** para la administración de la base de datos, y la instancia del motor de base de datos se ejecutó en un contenedor de **Docker**, facilitando la portabilidad y consistencia del entorno de la base de datos.
- **Comunicación:** Las reuniones y la coordinación del equipo se realizaron a través de **Discord**.

Arquitectura de la Aplicación

El proyecto se desarrolló siguiendo una arquitectura de tres capas.

Frontend (Capa de Presentación)

La interfaz de usuario es una aplicación web estática construida con **HTML5**, **CSS3** y **JavaScript**. Se encarga de presentar los datos al usuario y capturar sus interacciones. La comunicación con el backend se realiza mediante peticiones fetch a la API REST. La estructura del frontend se compone de varias páginas, cada una con una responsabilidad específica:

- login.html: Formulario de inicio de sesión.
- index.html: Página principal que lista los empleados y permite realizar acciones sobre ellos.
- insertar.html y actualizar.html: Formularios para crear y modificar empleados.
- detalle.html: Muestra la información completa de un empleado.
- movimientos.html y insertarMovimiento.html: Interfaces para visualizar y registrar movimientos de vacaciones.

Backend (Capa de Lógica)

El backend es una API REST desarrollada en **Python** utilizando el microframework **Flask**. Esta capa gestiona la lógica de negocio y actúa como intermediario entre el frontend y la base de datos.

- Utiliza **Flask-CORS** para gestionar las políticas de Intercambio de Recursos de Origen Cruzado (CORS), permitiendo que el frontend (servido desde un origen diferente) pueda consumir la API.
- La conexión con la base de datos se realiza a través de la librería **pyODBC**, que permite ejecutar los procedimientos almacenados.
- Expone varios endpoints para manejar las operaciones del sistema, como /login, /logout, /selectTodos, /insertarEmpleado, /actualizarEmpleado, /eliminarEmpleado, /movimientos, etc.

Base de Datos (Capa de Datos)

Se utilizó **Microsoft SQL Server** como motor de base de datos relacional. Toda la lógica de acceso y manipulación de datos está encapsulada en **procedimientos almacenados (SPs)** escritos en **T-SQL**. Este enfoque prohíbe el uso de SQL incrustado en el código del backend, aumentando la seguridad y el mantenimiento. Las tablas principales del sistema son Empleado, Puesto, Movimiento, Usuario, BitacoraEvento y Error.

Tecnologías Empleadas y Explicación

- **Frontend:**
  - **HTML:** Proporciona la estructura semántica de las páginas web.
  - **CSS:** Se encarga de los estilos visuales y el diseño responsivo para mejorar la experiencia de usuario.
  - **JavaScript:** Añade interactividad y gestiona la comunicación asíncrona con la API del backend.
- **Backend:**
  - **Python:** Lenguaje de programación principal para la lógica del servidor.
  - **Flask:** Microframework web para la creación de la API REST.
  - **pyODBC:** Librería que actúa como conector para interactuar con la base de datos SQL Server.
- **Base de Datos:**
  - **Microsoft SQL Server:** Sistema de gestión de base de datos relacional.
  - **T-SQL:** Lenguaje utilizado para escribir los procedimientos almacenados que contienen toda la lógica de la base de datos.
  - **Docker:** Plataforma de contenedores utilizada para ejecutar la instancia de SQL Server de forma aislada y reproducible.

Capítulo 3: Análisis de Resultados del Proyecto

A continuación, se presenta una tabla que evalúa el cumplimiento de los requerimientos funcionales del proyecto, de acuerdo con la rúbrica de evaluación.

Elemento Evaluado	Valoración	% de Implementación	Comentarios
Documentación y Diseño	Implementado en su mayoría bien	90%	La documentación cumple con la mayoría de requisitos. En la bitácora faltó documentación y el análisis de resultados están completos.
Base de Datos y	Implementado	100%	Se crearon todas las

Diseño	exitosamente		tablas según el modelo físico y se cargaron los datos desde el XML.
R1: Login, Logout	Implementado exitosamente	100%	El sistema valida credenciales, controla intentos fallidos, bloquea usuarios y registra todo en la bitácora.
R2: Listar Empleados y Filtro	Implementado exitosamente	100%	La interfaz principal lista todos los empleados y permite filtrar por nombre o documento de identidad.
R3: Insertar Empleado	Implementado exitosamente	100%	El formulario permite insertar nuevos empleados, validando que no existan duplicados por nombre o documento.
R4: ABC de Empleados	Implementado exitosamente	100%	Se pueden Consultar, Actualizar y Eliminar (borrado lógico) empleados desde la interfaz principal.
R5: Listar Movimientos	Implementado exitosamente	100%	Se muestra el historial de movimientos de vacaciones de cada empleado, ordenado por fecha descendente.
R6: Insertar Movimiento	Implementado exitosamente	100%	Se pueden registrar nuevos movimientos de vacaciones, validando que el saldo no resulte negativo.

R7: Trazabilidad (Bitácora)	Implementado en su mayoría bien	90%	Se registran casi todas las operaciones. Las únicas que no registra son las de login y buscar.
R8: Mensajes de Error	Implementado exitosamente	100%	Todos los errores controlados devuelven un código que es consultado en la tabla Error para mostrar un mensaje descriptivo al usuario.

Capítulo 4: Métricas del Proyecto

Métrica	**Total**	**Comentarios**
Horas trabajadas	~26h	No tuvimos mucho problema con errores. 2 horas para revisar todo antes de finalmente enviar la solución al profesor.
Número de sesiones de trabajo	3	Sesiones de trabajo en pareja.
Duración total de las pruebas (en horas)	3	N/A
Horas de resolución de problemas	3	Errores de autenticación al momento de acceder a la base de datos, tocó hacerla desde 0
Total de líneas de código	~1500	Sumando SQL, Python, HTML, CSS y JavaScript.
Número de commits en GitHub	38	Refleja un trabajo constante y colaborativo.
Cantidad de archivos	44	16 archivos SQL, 1 archivo Python, 7 archivos JS, 7 archivos HTML, 1 archivo

		CSS, y otros archivos de configuración y medios.
Cantidad de datos de prueba	100+	Registros cargados desde el archivo XML en varias tablas.
Cantidad de tablas creadas	9	Empleado, Puesto, Movimiento, TipoMovimiento, Usuario, BitacoraEvento, TipoEvento, DBError, Error.
Procedimientos almacenados	12	Cada funcionalidad de la base de datos está encapsulada en un SP.
Cantidad de funciones	11	Endpoints definidos en el backend para manejar las solicitudes del cliente. (No se cuentan las de Javascript)

Capítulo 5: Figuras Adicionales (GitHub)

Para evidenciar el trabajo colaborativo y constante, se pueden adjuntar gráficos generados por GitHub que muestren el historial de commits y las contribuciones de cada miembro del equipo.

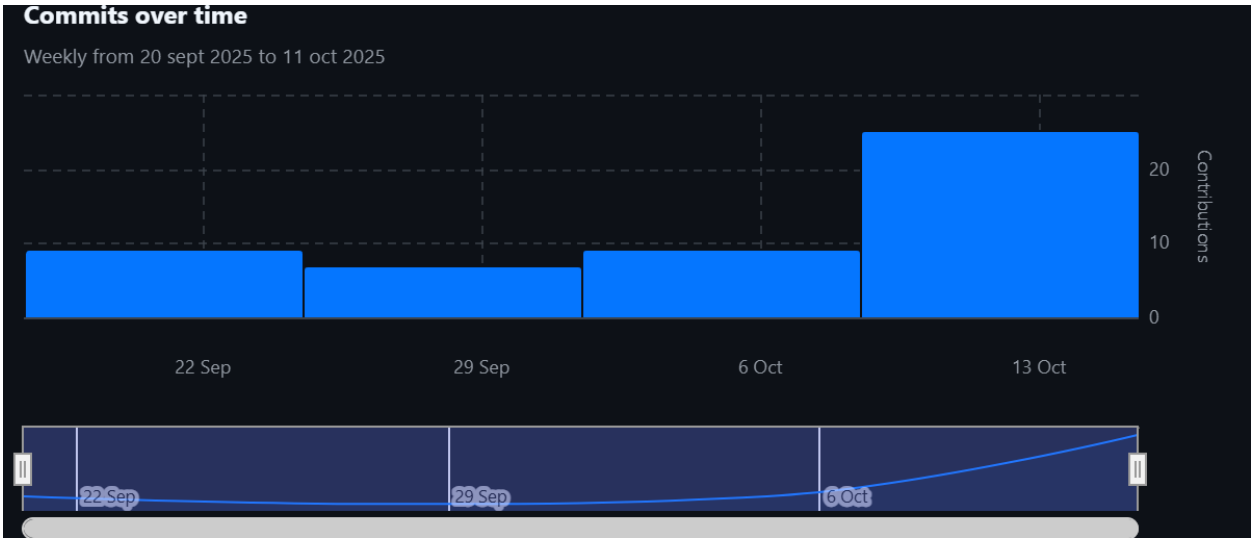

Figura 2: Gráfico de contribuciones del repositorio.

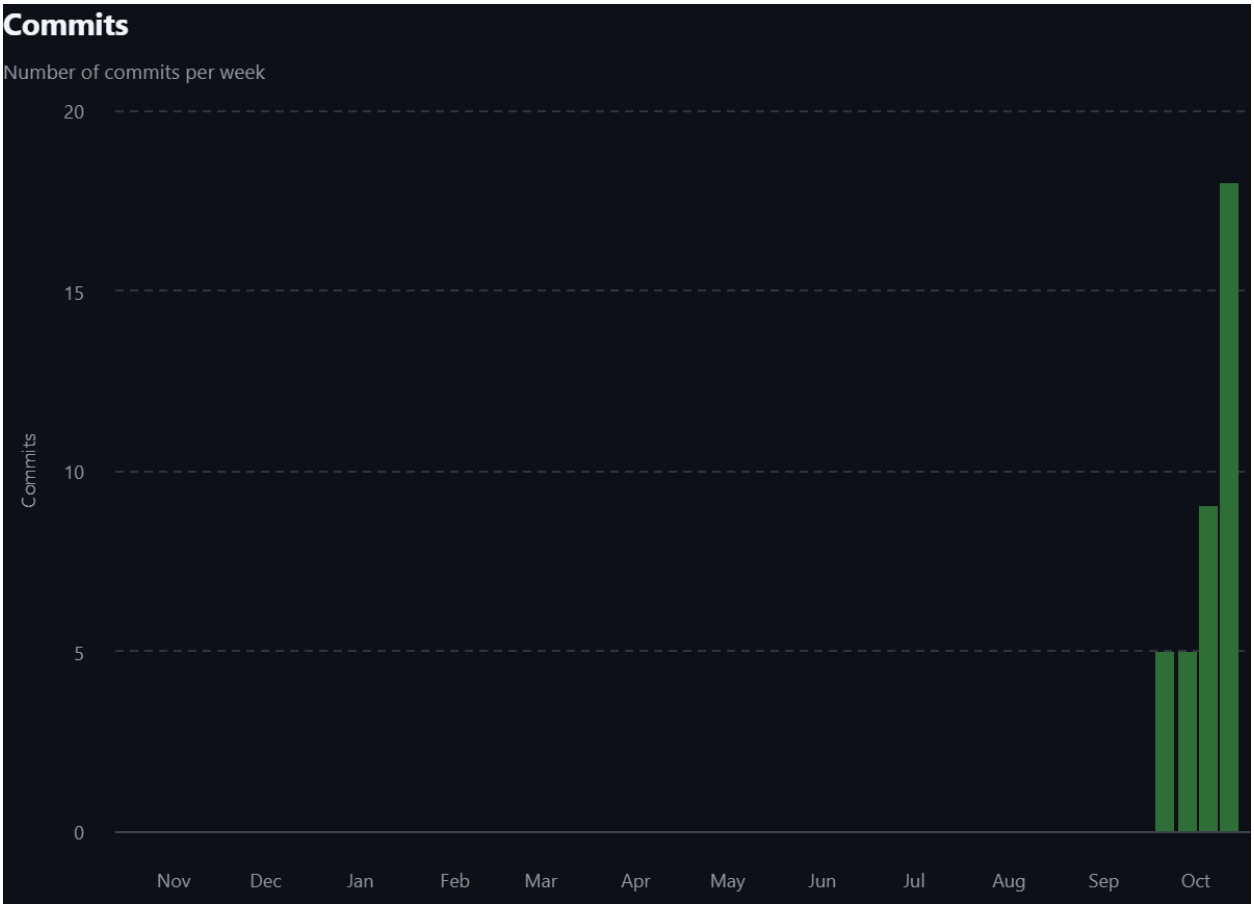

Figura 3: Gráfico de commits del repositorio.

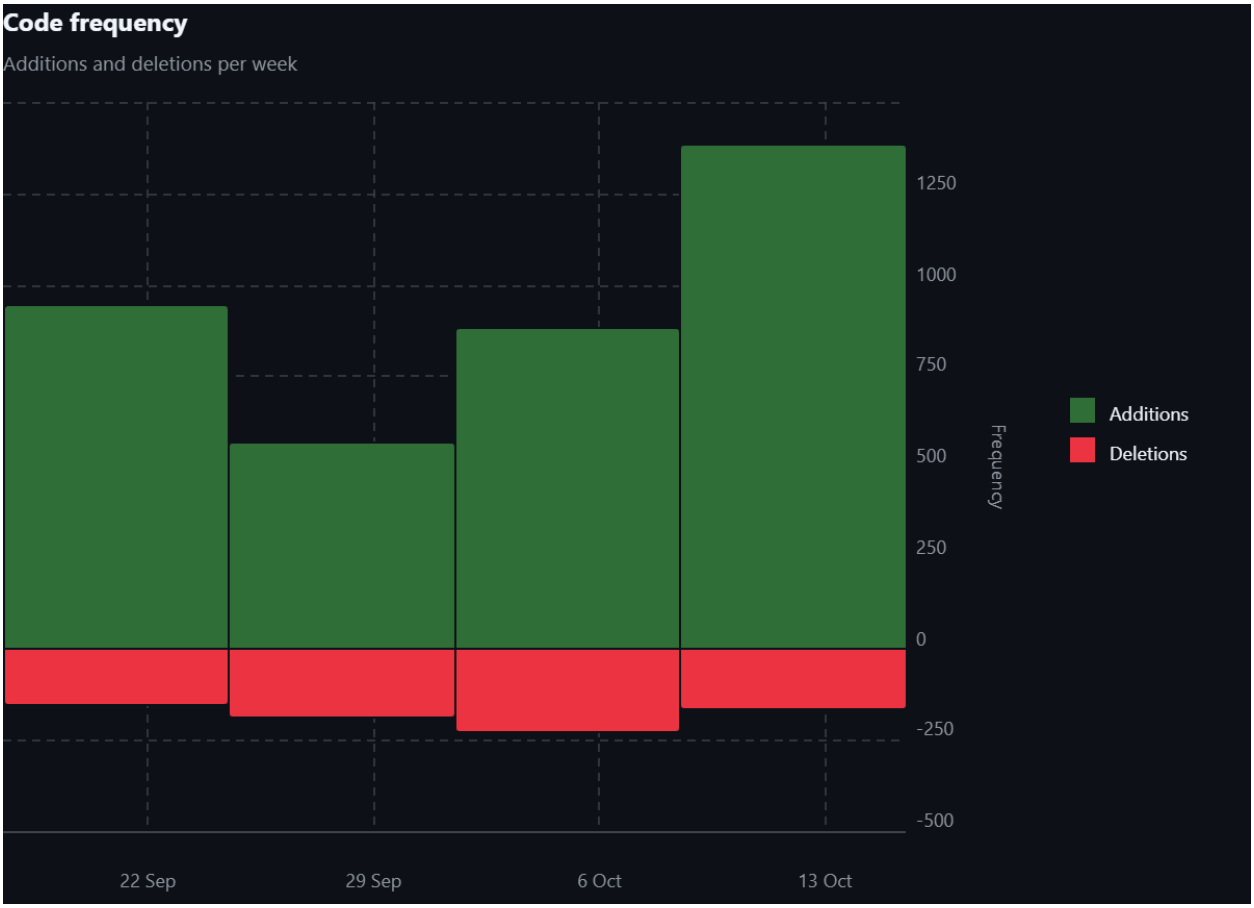

Figura 4: Gráfico de frecuencia de código del repositorio.

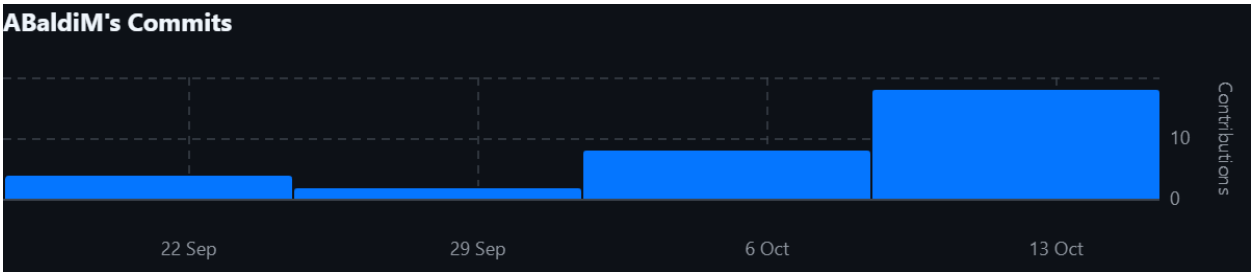

Figura 5: Gráfico de aportes de Andrés al repositorio.

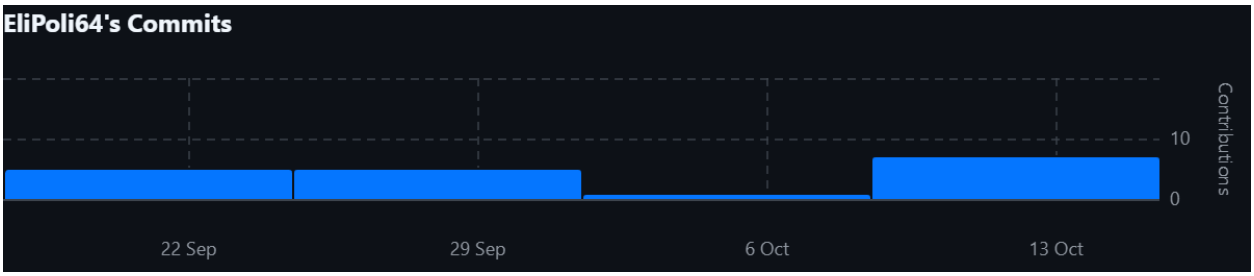

Figura 6: Gráfico de aportes de Elías al repositorio.